prior network activity, assigned priority information, service class levels, and the like. However, the Internet is essentially designed as a stateless interconnect mechanism to make the network more robust when faced with network failures and inconsistencies. Hence, Internet protocols and standards do not support, and actually make it difficult to share state information useful in providing enhanced services such as prioritization and quality of service management.

By way of distinction, user-level or application-level processes can readily exchange state information because these processes have control over creating and managing data structures needed to exchange such information. However, these processes must be specially programmed in client and server software to enable such behavior. Moreover, the mechanisms implemented by these processes cannot be readily interpreted by standard Internet infrastructure components. Therefore, behaviors such as prioritization and quality of service management are propagated through, but not implemented within the Internet infrastructure.

Internet standards implement a limited mechanism for the exchange of state information specified in RFC 2109 entitled HTTP STATE MANAGEMENT MECHANISM published by the Internet engineering task force in 1997. This standard specifies two HTTP headers called "set-cookie" and "cookie" that indicate an HTTP packet having state information contained in the payload portion. Browser software that recognizes these headers is enabled to extract the state information and save it in a local data structure referred to as a "cookie". The standard requires that each cookie be saved with an indication of the domain for which the cookie is valid.

installed, this dynamic domain assignment must work cooperatively with the public domain name system.

FIG. 3 illustrates a domain name server (DNS) redirection mechanism that illustrates how a client 205 is connected to a front-end 201. The DNS system is defined in a variety of Internet Engineering Task Force (IETF) documents such as RFC0883, RFC 1034 and RFC 1035 which are incorporated by reference herein. In a typical environment, a client 205 executes a browser 301, TCP/IP stack 303, and a resolver 305. For reasons of performance and packaging, browser 301, TCP/IP stack 303 and resolver 305 are often grouped together as routines within a single software product.

Browser 301 functions as a graphical user interface to implement user input/output (I/O) through monitor 311 and associated keyboard, mouse, or other user input device (not shown). Browser 301 is usually used as an interface for web-based applications, but may also be used as an interface for other applications such as email and network news, as well as special-purpose applications such as database access, telephony, and the like. Alternatively, a special-purpose user interface may be substituted for the more general-purpose browser 301 to handle a particular application.

TCP/IP stack 303 communicates with browser 301 to convert data between formats suitable for browser 301 and IP format suitable for Internet traffic. TCP/IP stack also implements a TCP protocol that manages transmission of packets between client 205 and an Internet service provider (ISP) or equivalent access point. IP protocol requires that each data packet include, among other things, an IP address identifying a destination node. In current implementations the IP address comprises a 32-bit value that identifies a particular Internet node. Non-IP